Exame de Linguagens de Programação II

11 Fevereiro 2005

I [6]

O objectivo deste exercício é desenvolver um interpretador para a linguagem IMP, definida pela seguinte gramática

```
Com ::= {proc} Id {() Id {}} Com {in} Com {end} |
      {var} Id {:=} Expr {in} Com {end} |
      Id {:=} Expr |
      {if} Expr {then} Com {else} Com {end} |
      {while} Expr {do} Com {end} |
      Com {;} Com |
      {println} {() Expr {}} |
      Id {() Expr {}}

Expr ::= Expr {+} Expr |
        Expr {<} Expr |
        Num |
        Id |
        {() Expr {}}
```

onde os símbolos terminais estão representados entre { } e Num e Id representam respectivamente os literais numéricos (ex: 0, 98) e os identificadores (ex, i, xpto, luis). Os programas nesta linguagem são comandos, gerados pela categoria sintáctica Com. Os únicos valores manipulados pela linguagem são os números inteiros, produzidos pela avaliação das expressões.

As construções imperativas são interpretadas da forma usual, sendo as condições do comando condicional if then else e do comando iterativo while do interpretadas como na linguagem C: "true" é representado por qualquer inteiro diferente de zero, e "false" é representado por zero. O comando proc permite declarar um procedimento local, o comando var permite declarar uma variável local.

Para simplificar, os procedimentos aceitam apenas um parâmetro. Dentro do procedimento, o seu parâmetro representa um valor inteiro, não uma variável de estado (tal como um parâmetro const em C).

A resolução de nomes na linguagem IMP usa a regra correcta de resolução estática.

O programa exemplo abaixo tem como efeito a impressão do valor $0 + 1 + 2 + \dots + 9 = 45$.

```
var res := 0 in
  proc Sum (x)
    var i:=0 in
      while (i < x) do i:=i+x end;
      res := i
    end
  in
    Sum(10); println(res)
end
end
```

- [1] Indique no programa exemplo acima qual é o âmbito da declaração dos identificadores Sum, res, i e x.

2. [1] Explique como representar a sintaxe abstracta da linguagem IMP usando um conjunto de classes na linguagem Java.
3. [4] Especifique a semântica operacional da linguagem IMP, programando em cada uma das classes que apresentou em (2) acima métodos de avaliação !

? evaluate(a: Environment)

de tal modo que, dada uma AST guardada na variável prog, a chamada

```
prog.evaluate(new Environment())
```

o execute completamente, ou assinale um erro de interpretação.

Importante: Environment implementa a interface "standard" usada nas aulas.

II [4]

1. [2] Para funcionar correctamente, o interpretador da linguagem IMP não precisa de utilizar fechos (*closures*) para representar os procedimentos, bastando representá-los usando pares *Abs(Id, Com)* que guardam apenas o nome do parâmetro e o corpo do procedimento. (Não se preocupe se utilizou fechos ou não no grupo anterior, é indiferente para a correcção da sua resposta).

Em geral, esta forma simplificada de representar valores funcionais (ou procedimentais) **não implementa** de forma correcta a regra de resolução de nomes estática.

Explique porque é que no caso da linguagem IMP tudo funciona bem, mesmo usando abstrações da forma *Abs(Id, Com)* em vez de fechos da forma *Clos(Id, Com, Env)*.

2. [2] Que erros de execução podem ocorrer durante a execução de um programa IMP? Discuta as vantagens de equipar a linguagem IMP com um sistema de tipos.

III [5]

No trabalho realizado nas aulas práticas, a linguagem de objectos OhOhFCT! é interpretada por tradução numa linguagem CORE mais básica, que permite manipular variáveis de estado, registos, e funções de ordem superior.

1. [3] Explique como implementar a linguagem IMP do mesmo modo, traduzindo de forma sistemática todos os programas IMP em programas CORE.

Para o efeito, considere **cada** uma das formas de comando e de expressão tal como aparece na gramática da grupo I, e mostre como cada uma pode ser representada fielmente na linguagem CORE.

Repare que existem 8 tipos de comando e 5 tipos de expressão.

2. [2] Considere a extensão IMPVAR da linguagem IMP com parâmetros passados por referência.

```
Com ::= {proc} Id {() {var} Id {}} Com {in} Com {end} |
      {proc} Id {() Id {}} Com {in} Com {end} |
      {var} Id {=} Expr {in} Com {end} |
      ...
```

Por exemplo, o programa

```
proc P(var x) x := 0 in var u := 2 in P(u); println(u) end end
```

imprime o valor 0, pois a variável u é passada por referência. Apresente um sistema de tipos para a linguagem IMPVAR que garanta que os seus programas não incorram em erros de execução.